**Version 1.0**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 30/mar/17 | 1.0 |  | Sorin Urcan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Security 4

2.4 Testability 4

2.5 Usability 4

3. Design Constraints 4

* **Introduction**

Uses Eclipse to design and implement a client-server application for the tax emitters and clients. The application should have two to three types of users (a regular user represented by the client, a tax mitter use who issues taxes and a system administrator) which have to provide a username and a password in order to use the application.

Data will be stored in a database. Using the Layers architectural pattern to organize the application. Using a domain logic pattern (transaction script or domain model) and a data source hybrid pattern plus a data source pute pattern most suitable for th e application.

* **Non-functional Requirements**
* *Update the displayed information each time an operation is executed, so that the viewed data is always up to date*
* *Usage of different Design Patterns: Layered Architecture, Transaction Script, Row Data Gateway and Server Pages with Servlets*
* **Availability**
* **Performance**
* **Security**
* **Testability**
* **Usability**
* **Design Constraints**

*[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]*